Aalto University
School of Science
Degree Programme of Computer Science and Engineering

Lodewijck Vogelzang

Cross platform development frameworks for start-ups.

Research for an e-learning platform, Lecter

Master's Thesis
Espoo, October 12, 2016

Supervisor: Professor Petri Vuorimaa
Instructor: Professor Petri Vuorimaa

Aalto University
School of Science
Degree Programme of Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author: Lodewijck Vogelzang

Title:
Cross platform development frameworks for start-ups. Research for an e-learning platform, Lecter

Date:	October 12, 2016	**Pages:**	vii + 51
Professorship:	WWW Applications	**Code:**	T-111
Supervisor:	Professor Petri Vuorimaa		
Instructor:	Professor Petri Vuorimaa		

There are solutions to target the vast majority of mobile platforms in one single project. The different strategies to develop an app for multiple platforms are compared based on a case study: Lecter. Lecter is going to be an e-learning platform that is accessible from mobile devices. At least iOS and Android are targeted, in order to target the vast majority of mobile platforms. Xamarin Forms, React Native and PhoneGap are three free frameworks to create an app for Android and iOS in a single project. Based on implemented prototypes of Lecter with use of the three mentioned frameworks, Xamarin is the closest to native development and PhoneGap is the closest to web development. When adopting cross-platform development, the decision between the frameworks affects the app development, the product performance and the business. Xamarin Forms is recommended to use for Lecter based on the skill set of the development team and the unlimited possibilities of Xamarin Forms.

Keywords: Cross platform development, Xamarin, React Native, PhoneGap, iOS, Android, Cloud Computing, AWS

Language: English

Acknowledgements

I would like to thank everyone who supported me during the two years of my double degree master's programme. My family and friends helped me to move forward and this Thesis, with guidance from Prof. Petri Vuorimaa, is the end of two wonderful years.

Espoo, October 12, 2016

Lodewijck Vogelzang

Glossary

Apache server	An effort to develop and maintain an open-source HTTP server.[1]
API	Application Programming Interface: a definition how external parties can utilize existing code or services.
AWS	Amazon Web Services: A public cloud services platform from Amazon.[2]
Bootstrap	A popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.[3]
CSRF attack	Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.[4]
Django	Django is a Python framework that makes it easier to build Web apps more quickly and with less code. It's free and open-source.[5]
HTML5	Hyper Text Markup Language: a markup language for describing web documents (web pages).[6]
HTTPS	HTTPS is a widely used Internet protocol for secured data transfer, based on the HTTP protocol [23].

[1] HTTPD: The Apache HTTP Server Project, 1st September 2016: https://httpd.apache.org

[2] AWS: Cloud Computing with Amazon Web Services, 1st September 2016: https://aws.amazon.com

[3] Bootstrap: The world's most popular mobile-first and responsive front-end framework, 1st September 2016: http://getbootstrap.com

[4] OWASP: Cross-Site Request Forgery (CSRF), 1st September 2016: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

[5] Django: The Web framework for perfectionists with deadlines, 1st September 2016: https://www.djangoproject.com

[6] HTML introduction, 1st September 2016: http://www.w3schools.com/html/html_intro.asp

LMS	Learning Management System, providing trainings and tracking users are for instance typical functions of an LMS.
Postgresql	A robust, open-source database implementation that conforms to the current SQL standard.[7]
Python	A popular open-source programming language that runs on many platforms.[8]
REST	REST is a mode of thought for service abstraction. It helps to truly understand the original look of HTTP and fully utilize current Web features [10].
Route53	A highly available and scalable cloud Domain Name System (DNS) web service.[9]
SCORM	SCORM is a set of technical standards for e-learning software products.[10]
SQL injections	Injection of a SQL query via the input data from the client to the application, to access or modify a database without permission.[11]
TLS certificate	A certificate provided by a trusted 3rd party, to ensure valid HTTPS connections for a server.[12]
XSS attacks	Injections in which malicious scripts are injected into trusted web sites.[13]
XML	Extensible Markup Language. Human- and machine-readable language, designed to store and transport data.[14]
XAML	Extensible Application Markup Language: an XML-based markup language to define user interfaces, developed by Microsoft.[15]

[7]PostgreSQL: The world's most advanced open-source database: https://www.postgresql.org

[8]About Python, 1st September 2016: https://www.python.org

[9]Amazon Route 53, Domain Name Server, DNS Service, 1st September 2016: https://aws.amazon.com/route53/

[10]SCORM explained, 1st September 2016: http://scorm.com/scorm-explained/

[11]SQL Inection, 2nd September 2016: https://www.owasp.org/index.php/SQL_Injection

[12]What is SSL, TLS and HTTPS? 2nd September 2016: https://www.symantec.com/page.jsp?id=ssl-information-center

[13]Cross-site Scripting (XSS), 2nd September 2016: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

[14]XMLhttp://www.w3schools.com/xml/

[15]Microsoft: What is XAML? 2nd September 2016: https://msdn.microsoft.com/en-us/library/cc295302.aspx

Contents

Chapter 1

Introduction

Developers have to decide which platforms to target before building an application. Back in 2012, Windows was undeniably the most popular operating system in use. An application targeted for Windows XP, Vista and 7 would cover about 90% of the global market for software applications[1]. This market could be reached with a single application, written in one of the programming languages that are supported by all Windows versions.

Nowadays the diversity of available platforms has increased as the mobile systems have gained a bigger market share. Mobile platforms are currently roughly as popular as desktop platforms for web browsing[2], one of the easiest ways of accessing software. Since the increased diversity of the available platforms, new strategies are needed to target the majority of the market. There are multiple strategies to reach the goal of targeting a majority of the market and this Thesis will compare these strategies.

The different strategies will be compared based on a case study: Lecter, an e-learning platform under development. The back-end system is implemented and running in a public cloud. The front-end system remains to be implemented. A detailed description of Lecter will be given in the chapter 2: Context, including a business description and the technical specifications. One of the requirements of the front-end for Lecter is that it needs to be supported by the popular operating systems for mobile devices.

[1]Market share held by the leading computer operating systems worldwide from January 2012 to December 2015, http://www.statista.com/statistics/268237/global-market-share-held-by-operating-systems-since-2009/, Accessed: 06.04.2016

[2]Platform popularity statistics for web accesses, http://gs.statcounter.com/#all-comparison-ww-monthly-201503-201603, Accessed: 10.04.2016

The main goal of this research is to determine the best strategy to develop Lecter's front-end. By determining the differences between the suitable strategies, I will find and recommend the best option. The research question will be stated as:

- What are the differences between the suitable cross-platform mobile development environments for Lecter's front-end development?
  - What are the impacts on the development that come with each environment?
  - What are the impacts on the product that come with each environment?
  - What are the impacts on Lecter's business model that come with each environment?

Facebook's React Native, Microsoft's Xamarin and Adobe's PhoneGap are big projects to provide a way to build cross platform apps. QT is an independent solution to build cross platform apps. Open standard HTML5 is a solution to build cross platform apps which will run in a browser. There are many more solutions on the market besides these popular tools, that will be discussed briefly in the chapter research methods.

This Thesis will research and compare React Native, Xamarin and PhoneGap. These solutions for creating a cross platform app are selected based on the criteria for Lecter's front-end. A Prototype will be implemented using the different solutions in order to be able to compare React Native, Xamarin and PhoneGap. The prototypes' quality will be measured with several metrics, described in chapter 5: Research Methods. The prototype's quality is defined based on the technical performance. The User Experience (UX) design will be basic and according to the platform specific standards. A more detailed UX will be designed by an external party in a later stage of the development process and is not part of the Thesis.

The outcome of this research will give a detailed overview of the selected technologies. The research data will indicate the consequences of choosing one of the technologies in the case of Lecter. The consequences will be generalized as far as possible to provide useful information for developers who are at the point of choosing a solution to create a cross platform app.

Chapter 2

Context

Lecter is a project that started when clients of Lovo Creations[1] asked for new e-learning solutions. Lovo Creations is specialized in creating tailored e-learning software for Dutch manufacturing plants. As e-learning could be applied in many industries, customers covered different markets:

- Food industry (Zeelandia, LambWeston, Special Fruit)
- Packaging (Elopak)
- Adhesives (Bison)
- Biological pest control (Koppert)
- ...

The created e-learning tools were generally based on existing corporate training materials. The digitalized version included new media items such as images and videos. In combination with new animations and exercises, the trainings became more interactive. The general structure of the created e-learning tools is visualized in figure 2.1. Usually, a training starts with a front page and an introduction explaining the goal of the training. The main part consists of chapters including slides with text, animations, images and videos. The chapters possibly include simple questions or exercises to make the training interactive. The results of those exercises are not saved. An official exam comes at the end of the training after the chapters. The exam starts with a short explanation about the exam and the trainee will be asked to enter his/her details. These details are used to store the score of the trainee. This happens automatically when the results are calculated and

[1] Lovo Creations, Dutch one-man business by Lodewijck Vogelzang, http://www.lovocreations.nl/

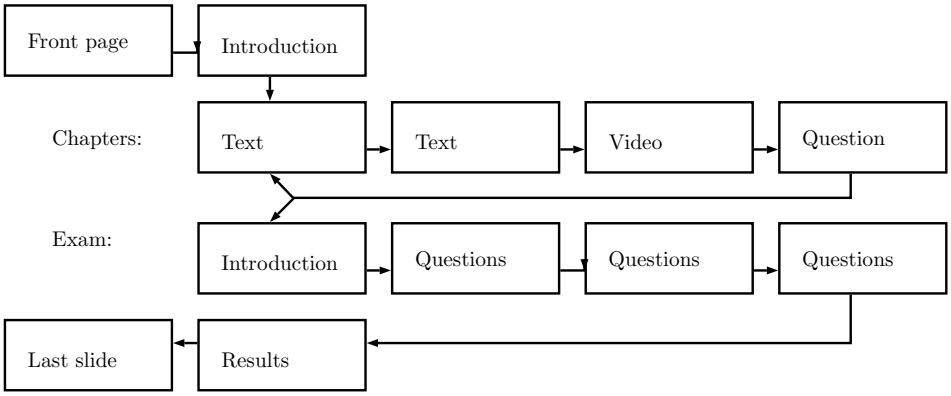


Figure 2.1: General e-learning structure.

shown.

The current solution does not answer the new needs of companies. The used technology, Adobe Flash, does not support mobile devices and is losing support for more platforms (and that is a good thing [13]). Apart from business reasons from the perspective of Lovo Creations, this was one of the reasons to start a new project and resulted in the Lecter project.

The widely used tool to describe a business, the Business Model Canvas [25], is used to give a complete overview of the business idea in section 2.1. Lecter's competitors will be listed in section 2.2. An overview of the final product will be given in section 2.3, with a technical description of the back-end of Lecter that already exists in section 2.4.

Figure 2.2: Logo of lecter.

2.1 Business case

The nine blocks of the Business Model Canvas together give a comprehensive view on Lecter. The Business Model Canvas is visually represented in appendix A.

Lecter offers an effective education system for industrial companies by providing a way to easily create and maintain e-learning trainings. The advantages of e-learning, explained in section 3.1, are the main incentives for companies to replace a selection of existing classical trainings to e-learning trainings. The advantage of Lecter compared to outsourcing the creation and maintenance of e-learning tools, is that the trainings will be more up to date. Updating a training should be easy. A spelling correction in a slide should be a one minute task. Even replacing a picture should be conceivable as a one minute task.

Lecter makes use of the opportunities of Web 2.0. Publication [22] describes the transition from Web 1.0 to Web 2.0: "Web 2.0 harnesses the Web in a more interactive and collaborative manner, emphasizing peers' social interaction and collective intelligence, and presents new opportunities for leveraging the Web and engaging its users more effectively." This concept applies to Lecter, as people can create, share, remix, repurpose and pass along training materials. The term e-learning 2.0 is used to describe the integration of e-learning with Web 2.0 technologies. A learning platform with a community is an example of e-learning 2.0 [8].

The targeted customers, the clients who might buy or license Lecter, are industrial companies like the former clients of Lovo Creations. Examples of the individuals who are targeted in companies are Human Resources managers, Continuous Improvement managers and Safety managers. These people should be convinced in buying Lecter and will become the people who use Lecter. How lecter could be used is described in a user story:

The safety manager of a factory has the responsibility to educate all employees about the safety rules in the factory. When an accident happens caused by people, the safety manager has to prove that a lack of knowledge about safety rules was not the cause of the accident. Trainings already exist in the form of a printed text document, created by the safety manager, and all people who enter the factory have to read this training in order to gain access. These documents are going to be digitalized by the manager using Lecter on

a tablet. The text from the former training is entered in a Lecter training and pictures are taken and added to the training in the factory. Everyone who enters the factory has to take the tablet and do the training. As there is no longer a manager needed to check the training's results, contractors can gain access to the factory in the weekends and nights as well. In case of an accident, the managers has access to the people who did the trainings as a proof that everybody is up-to-date about the safety regulations.

2.2 Competitors

Many e-learning solutions are already published. E-learning solutions fit usually in one of the following three descriptions:

- Outsourced customized e-learning
- Offline e-learning editors
- Local network e-learning platforms
- Online e-learning platforms

The first approach to obtain customized e-learning materials is to hire specialized companies. Specialized companies can create high-quality trainings. This is a very easy approach but usually expensive and the maintainability of the materials is low. Companies that provide a service to create e-learning trainings are indirect competitors.

Offline e-learning editors are tools to create trainings locally. Microsoft PowerPoint is the simplest example where trainings can be created offline. A local file, the PowerPoint presentation, is the output of the e-learning creation process. There are more sophisticated solutions designed to create e-learning materials specifically. Adobe Captivate[2], Articulate Storyline[3] and iSpring[4] are examples of local e-learning creation tools. Offline e-learning editors are generally SCORM compliant. SCORM is a technical e-learning standard to

[2]Adobe Captivate: Create any kind of eLearning end-to-end. For any device. 29th August 2016: http://www.adobe.com/uk/products/captivate.html

[3]Articulate Storyline 2: Create interactive e-learning, easily. 29th August 2016: https://www.articulate.com/products/storyline-why.php

[4]iSpring: E-Learning Software That Really Works! 29th August 2016: http://www.ispringsolutions.com

stimulate the rapid development of learning repositories [17]. This standard simplifies the implementation of different kind of training materials into a Learning Management System (LMS).

Local network e-learning platforms are more popular nowadays. The big advantage of local network e-learning platforms is that it is usually encapsulated in an LMS. The platforms are not used to create materials only, but to manage the trainings and users at a higher level as well. These platforms are generally more secure because they are not accessible from outside the company. An installation on a local server is required, so the drawback of local network e-learning platforms is the relatively complicated setup. Moodle[5] is an example of a widely used local network e-learning platform. Chamilo[6] is another open-source example. These products are the main competitors of Lecter, even though Lecter is an online solution.

Online e-learning solutions are in general similar to local network e-learning platforms. The security is, however, harder to achieve, but there is no installation needed. Another big difference is the ability to share trainings with a global community. Lynda[7] is an example of such a platform. Lecter will also belong to this category and Lynda would be a direct competitor of Lecter.

From the user perspective, Lecter is going to be as open as Lynda. Users can find trainings easily without logging in, on multiple platform including mobile platforms. Lecter provides, unlike Lynda, also free materials and Lecter targets a smaller group of customers. Lecter only aims for companies and corporate trainings, and therefore Lecter can be designed for this niche market specifically. The trainings are less customizable but easier to create. Only certain types of exercises are available from templates, to make the creation process of e-learning materials easier and more guided to a professional and effective training.

[5]Moodle: Open-source learning platform, 30th August 2016: https://moodle.org

[6]Chamilo LMS: e-learning that adapts to your needs, 30th August 2016: https://chamilo.org

[7]Lynda: Learn a New Skill Online, on Your Time, 30th August 2016: https://www.lynda.com

Udemy[8], KhanAcademy[9], edX[10] and Coursera[11] are other platforms that are like Lynda competitors of Lecter. The main difference is the targeted market which results in different revenue models and training structures.

2.3 Product description

Lecter should be very accessible for new users. Users should be able to access trainings without installing software or logging in. Youtube is an example of a platform where the contents, videos in this case, are accessible from the front page and without logging in.

Lecter is not going to have a sophisticated editor. The user is responsible for entering information as text and media items. Lecter is responsible for building an effective e-learning training out of that, visually optimized for PCs and mobile devices.

The user experience (UX) design will not be discussed in this Thesis. It is an important part of the software development for Lecter, but out of the scope of this Thesis. The UX design will be outsourced after a proof of concept is created for Lecter.

One of the main requirements is that Lecter should be accessible from a wide range of devices. Lecter will be used on desktops (to copy materials easily from existing trainings into Lecter) and on tablets (to take and insert pictures on the job). Lecter should support the major operating systems and share the trainings over all the supported operating systems. This makes it possible to switch between platforms when a user creates a training and users working with different platforms can share their trainings.

Lecter has to share and download trainings online, so it requires an Internet connection. It should also be possible, though, to do a training in an area without Internet connection.

[8]Udemy: Learn Anything, On Your Schedule, 30th August 2016: https://www.udemy.com

[9]Khan Academy; You only have to know one thing: You can learn anything. For free. For everyone. Forever. 30th August 2016: https://www.khanacademy.org

[10]edX: Free online courses from the world's best universities, 30 August 2016: https://www.edx.org

[11]Coursera: Free Online Courses From Top Universities, 30th August 2016: https://www.coursera.org

The development time and costs are other restrictions. To be able to test a real prototype or sell a basic version of the product, low cost, fast development is required. Freedom to customize the software without limits is not required, but it should be possible at a later stage when time and money are less important. This is generally a valid requirement for early stage start-ups.

2.4 Back-end description

The backend of Lecter is running in Amazon's public cloud. Amazon Web Services (AWS) provides dozens of tools to develop web applications. Figure 2.3 represents how Lecter's backend is composed of AWS building blocks. Abbrevations and technical terms are explained in the glossary of this Thesis.

Django is used to create a REST API and a simple web interface. Django projects are strictly structured and written in Python and HTML5. The predefined structure involves common patterns to protect against malicious attacks. The documentation[12] explains how Django projects should be protected against XSS attacks, CSRF attack, SQL injections and other common attacks.

An Apache server runs the Django project in the Amazon Cloud. This server could only be reached by using a secure (HTTPS) connection for security reasons. The required certificates to enable secure connections are generated automatically. Let's Encrypt[13] is an open-source project to obtain free TLS certificates and is used for Lecter.

Amazon provides a tool to register domain names: Route53. Route53 contains a DNS as well and Lecter makes use of this to redirect users to a running server. Lecter will run on multiple servers to improve availability and scalability. A load balancer will be used to spread the workload, as visualized in figure 2.3. User data will be stored in a PostgreSQL database, except the videos and images. These media items, even the media uploaded by users, are stored in S3.

[12]Django security documentation, 7th June 2016: https://docs.djangoproject.com/ja/1.9/topics/security/

[13]Let's Encrypt, a free, automated, and open certificate authority: https://letsencrypt.org

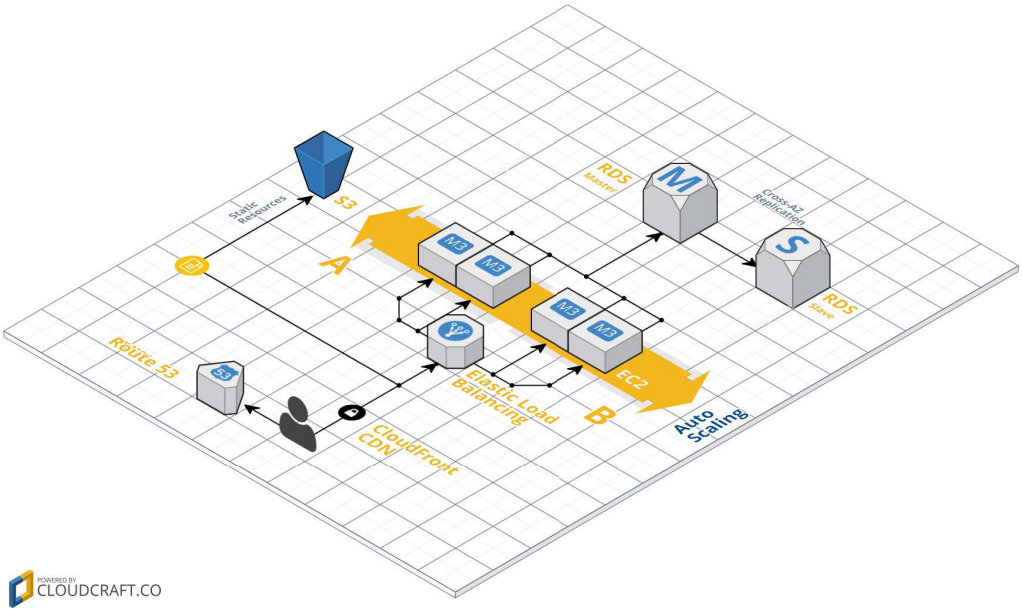


Figure 2.3: Backend design of Lecter

Chapter 3

E-learning

This chapter will describe a part of the background of the research. A detailed description of the context, Lecter, is provided in chapter 2. As Lecter will become an e-learning platform, a background on e-learning will be discussed in this chapter.

3.1 Definition

E-learning is a widely used term. The term is well known in the sense that many people have associations with it, even though the explicit definitions in literature vary. I would like to use the definition that is used in [4]: "We define e-learning as instruction delivered on a digital device such as a computer or mobile device that is intended to support learning." So electronic devices are used for educational purposes and the Internet is not necessarily involved.

The opposite of e-learning would be training methods that do not make use of electronic devices. Typically Face-to-face learning and paper-based materials are used.

Already in the mid 60s, researchers tried to educate people using machines that were the precursors of the computer [32]. E-learning is nowadays widely implemented in different sectors as corporate organizations, educational institutions and in the public sector [19, 24, 33]. As this Thesis is related to a business case that targets the corporate organizations, e-learning is researched and reviewed for this sector only.

3.2 Implementing e-learning in corporate organizations

The financial aspects are often mentioned as one of the benefits of e-learning. It is possible to implement e-learning in a cost efficient way, but traditional learning could be cost efficient as well. Publication [30] aimed to define a formula to calculate the return on investment for e-learning projects. Among many other factors, these costs are taken into account: crafting the digital content, preserving the content, hosting an LMS, equipment and promotion. The total costs of implementing e-learning is able to be calculated with help of the presented models.

Before implementing, the readiness could be assessed [15]. Whether e-learning could be implemented depends on the used software, but probably even more on the people who organize the e-learning implementation. Parameters that affect the success of e-learning implementations researched [14, 31]. Guidelines and consultants are available nowadays to help implementing e-learning, because many companies started using e-learning.

3.3 Advantages and drawbacks

The advantages of e-learning compared to classical learning, from both the trainee's and the company's perspective [2, 35], are:

- **Self paced**: The trainees can do an e-learning training alone so there are no time limits. Trainees can take extra time when the training is perceived as complicated, or take extra time when they want to look for more background information. This is an advantage compared to classical learning in case there is a teacher involved.
- **Always available**: Access to a computer is the only requirement to be able to do an e-learning training. This enables trainees to do trainings at any time and at any location.
- **Up-to-date content**: Small updates, for instance updating some text or inserting a new picture, do not require the training to be redistributed in a time consuming way. Updates can be shared automatically when an Internet connection is available. Accessing outdated materials is in that case impossible after an update is performed.

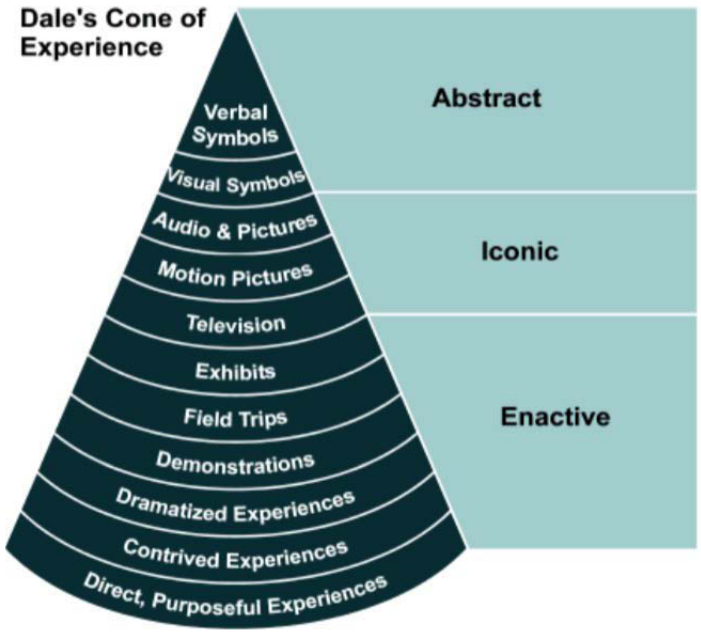


Figure 3.1: Dale's Cone of Experience [6]

- **Interactive learning**: Videos, images and exercises can improve the learning process [34]. As [28] interprets Dale's Cone of Experience (figure 3.1): "What he really said was that learning becomes more meaningful when abstract learning and concrete experience are related."
- **Globally consistent content**: Updates are done electronically and so is the distribution of the updates. Training contents remain consistent at all locations that access the training.
- **Personalized training**: It is easier for e-learning trainings, compared to printed materials, to respond to the trainee's preferences. Based on user input as a profile or selected preferences, an e-learning training can provide contents tailored for the trainee. By providing only the necessary contents, the trainee's workload decreases and the effectiveness of the training increases. Trainees can optionally access in-depth information for relevant subjects. Several personalization strategies are assessed in [9].
- **Automatically tracking**: Data could be saved automatically. This provides a way to analyze the trainees and the trainings. Manual registration is obsolete as training results could be documented automatically, and therefore available at any time.

- **Reduce carbon footprint**: As explained in [1], e-learning is an example of Green Computing. Computers or tablets are used to avoid wasting paper.
- **Low costs**: E-learning could be cost-efficient when it replaces human labour as teaching, distributing or documenting. This is in practice a prominent reason to implement e-learning. The initial costs to setup an e-learning environment is a long term investment. Practical examples are given in [35], section 3.2 describes the financial impacts in more details.

Publication [2] shows the disadvantages of e-learning as well. This research contains an online survey for employees of a well-known bank in Europe. The participants pointed out some advantages for classical learning compared to e-learning: "Face-to-face training continues to be perceived as a more motivating methodology compared to virtuality and with better explanations from the course trainers."

Advantages of classical learning:

- **More motivating**: Trainees pointed out to be more motivated when they belong to a group. Additionally, an invitation for a group training is perceived as an opportunity offered by the company to progress in their career.
- **Better explanations**: A teacher can use more words for explanations and the communication between trainer and trainee is reciprocal. Problem solving is easier with a human trainer as problems can be discussed verbally.
- **Peer collaboration/discussion**: Group trainings allow discussions between participants. Discussions can increase motivation and enhance learning.

Especially the organization benefits from e-learning. From the company's perspective there are no important drawbacks except the trainee's perception and the initial costs. As [2] concludes it's research: "Such results state that while the benefits of distance methodology can be clearly identified from the company's point of view (i.e., as a flexible and efficient methodology to develop the employees' skills and knowledge), from the employees' standpoint, the advantages of virtual training are not so clear and depend to a great extent on their attitude towards the use of virtuality." It is, however, researched how to improve the trainee's perception of e-learning trainings.

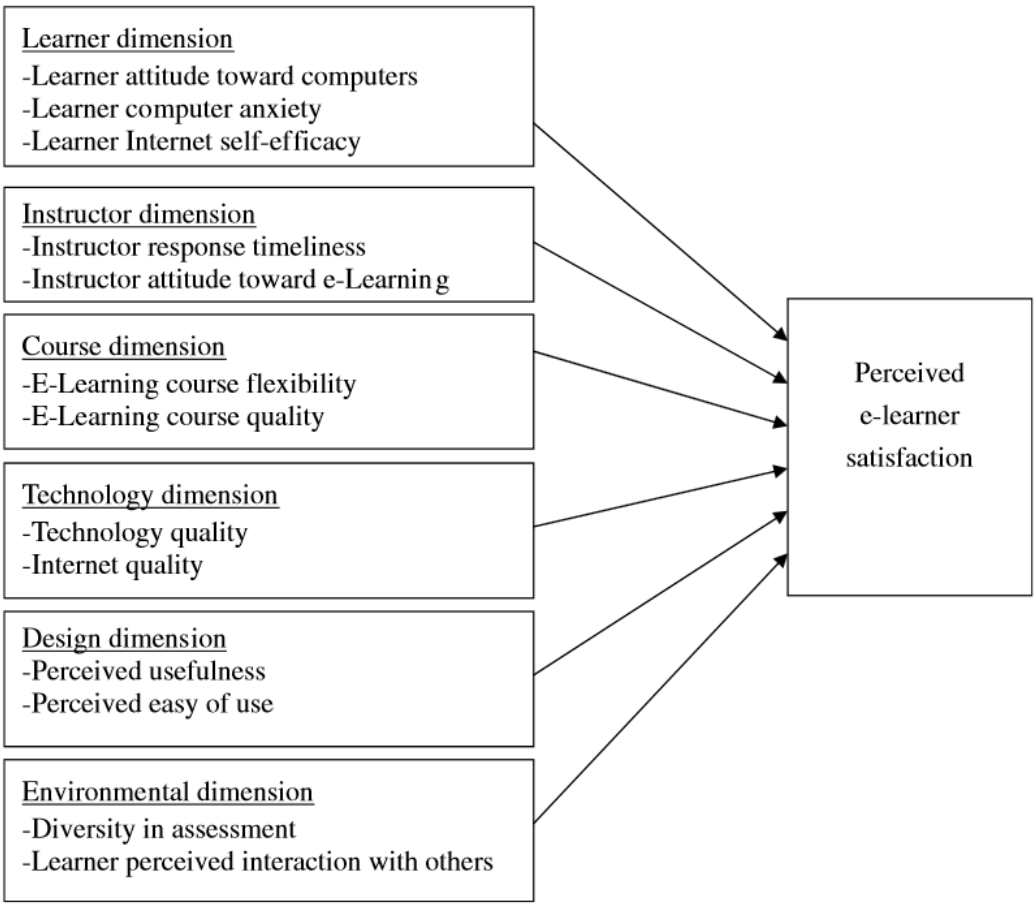


Figure 3.2: Critical factors that affect learners' satisfaction [31]

3.4 Success factors

Multimedia learning is researched and found out to be effective if used properly. A cognitive overload affects the learning negatively, as explained in [20]. This paper determined three different types of cognitive loads: essential processing, incidental processing and representational holding. The publication provides examples to reduce the different kinds of cognitive load in multimedia learning. For instance, narrated text is easier to process than visual text in a video and background music makes a video harder to process. All the mentioned principles in [20] will be taken into account during the design of Lecter to create effective learning materials.

Publication [31] points at the critical factors that affect learners' satisfaction. The learners' satisfaction is the main metric to assess the success of e-learning according to this publication. Figure 3.2 shows the specified dimensions, including factors that are not related to the e-learning tool as a product. As these external dimensions will be important as well to the perception of Lecter, a service to help implementing e-learning in a corporate organization would improve the customers' satisfaction.

Research in educational organizations resulted in similar conclusions. Publication [36] recommends schools to take into account external factors while designing a virtual learning environment: "A well-designed system might reduce the learners' frustration and attract them to continue using the system. Teaching system design should not focus only on the technical aspects. There is more to take into consideration."

Chapter 4

Cross platform frameworks

Lecter's description in chapter 2 includes a description of the front-end that does not exist yet. In this Thesis, I will research the different ways to implement the front-end. The various methods and existing means to implement the missing front-end of Lecter are discussed in this chapter.

"Write once, run anywhere" was the promise of Java at the first release in 1996 [5]. Java was designed to run in a virtual machine so running Java programs became platform independent. At present time, Swift is a new open-source programming language in development that targets the same principle: write once, run anywhere. The goal of the development of Swift is explained in the official documentation: "Our goal is to provide source compatibility for Swift across all platforms, even though the actual implementation mechanisms may differ from one platform to the next."[1].

The development of Java and Swift show the demand for cross-platform development tools. These projects are, however, not suitable to target mobile development because mobile platforms have platform specific restrictions. For instance, iOS, Android and Windows use different user interface frameworks [18]. A visual text item is represented by a UITextView on iOS, an EditText on Android and a TextBox on Windows. There are tools developed and in development to overcome this problem. This chapter will discuss these tools.

[1]Swift documentation about platform support, 8th June 2016: https://swift.org/about/#platform-support

4.1 Free solutions

Xamarin is one of the tools in development to create cross-platform apps with a shared codebase in C#. Xamarin was founded in 2011 and acquired by Microsoft in 2016. Microsoft relicensed Xamarin as open-source under the MIT license and aims to have over 1.4 million developers[2]. Xamarin's promises cross-platform native app development with a shared codebase: "Build native apps for multiple platforms on a shared C# codebase. Use the same IDE, language, and APIs everywhere."[3]

Facebook released React Native in 2013, with a similar goal as Xamarin: "The focus of React Native is on developer efficiency across all the platforms you care about - learn once, write anywhere."[4] React Native is a based on Facebook's JavaScript library React. This library is designed to program according to the flux architecture: an architecture with unidirectional data streams. React Native exports JavaScript projects to native apps for iOS and Android.

PhoneGap is the multi-platform development kit from Adobe: "Create experiences for multiple platforms with a single codebase so you can reach your audience no matter their device."[5] Unlike Xamarin and React Native, PhoneGap does not export projects to native apps. PhoneGap provides open-source libraries to create apps in HTML5 and run it in a way that looks like a native app.

4.2 Commercial solutions

Since 1995 QT develops tools to create cross-platform applications with a shared codebase: "Write your source code once and run it anywhere on any device."[6] QT is available under a open-source license, but only for projects that will be published as open-source. To develop commercial applications with QT, a licensed version of QT is required. Applications will run na-

[2]Microsoft released Xamarin for free, 31st March 2016: http://techreport.com/news/29929/xamarin-now-comes-free-with-visual-studio

[3]Xamarin homepage, 8th June 2016: https://www.xamarin.com

[4]React Native documentation: https://facebook.github.io/react-native/

[5]PhoneGap's home page, 8th June 2016: http://phonegap.com

[6]https://www.qt.io

tively on desktop, mobile and embedded platforms. Appcelerator is an other commercial solution that can be used to create cross-platform apps that run natively, but Appcelerator supports mobile platforms only.

Alpha Anywhere, Kony and Sencha are examples of commercial solutions that support cross-platform development by using web technologies. Alpha Anywhere is a commercial environment for "building native-quality, cross platform web and mobile business applications."[7] Kony "enables you to easily develop and deploy apps across channels from a single code base."[8] and Sencha's goal is to "Increase development efficiency by developing once for multiple platforms and devices."[9]

4.3 Alternative solutions

The latest web technologies provide ways to create web apps that are accessible for all major browsers. Web-apps can target many different platforms with one shared code base but have limitations. Web-apps do not run natively but in a browser and web-apps have therefore limited access to native APIs [3]. Besides that, native apps usually consist of compiled code, which is faster than interpreted languages such as JavaScript. On the other hand, interpreted languages provide significantly higher programmer productivity and software reuse [26]. Native apps and web apps are both popular but a battle between the two different strategies will perhaps end with one winner [21]. Others believe that the different strategies, including a hybrid option, will become equally popular [29].

Certain cross-platform development tools are developed to create games only. Cocos2s, Corona and Unity 3d Labs are examples of frameworks to build games and target multiple platforms with a single code base.

Other tools provide a way to create cross-platform apps by providing customizable templates. In many cases, it is possible to create an app without writing a single line of code. These tools are powerful to create simple apps, but the templates are limited. Examples of template based tools are: Appery.io, AppMachine, AppMakr, Appsbar, AppsBuilder, AppyPie, BiznessApps, BuildFire, Fliplet, GoodBarber, MobileRoadie, Shoutem.

[7] Alpha software home page, 9th June 2016: http://www.alphasoftware.com
[8] Kony website, "Why Kony", 9th June 2016: http://www.kony.com/about/why-kony
[9] Sencha home page, 10th June 2016: https://www.sencha.com

Chapter 5

Research methods

As described in section 2.3, Lecter should support both desktop and mobile platforms. Web technologies will be used to target the major desktop platforms: Windows and macOS. Recent versions of Windows (Windows 7 and later) and macOS cover around 85% of the market share in 2015[1]. To target the vast majority of mobile devices, at least Android and iOS should be supported. Android and iOS combined represent about 99% of the market in 2016[2].

To speed up the product development, the mobile platforms could be targeted with a single project. Providing a web app would make the product accessible for mobile devices, but has limitations that native apps do not have. Building native apps for the major mobile operating systems in a single project is the solution and many frameworks are recently built to support this.

Figure 5.1 shows the different development options. This diagram was published in [7] and does not take into account the current frameworks to develop cross-platform native apps. These frameworks aim to have the pros of native apps according to the diagram, but without the cons.

The targeted platforms for Lecter are Android and iOS. From the solutions that are mentioned in section 4, three open-source projects will be compared. Xamarin, React Native and PhoneGap are widely used environments and free

[1] Global market share of Windows 7, December 2015: http://www.statista.com/statistics/218089/global-market-share-of-windows-7/

[2] Global market share held by smartphone operating systems, 2016: http://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

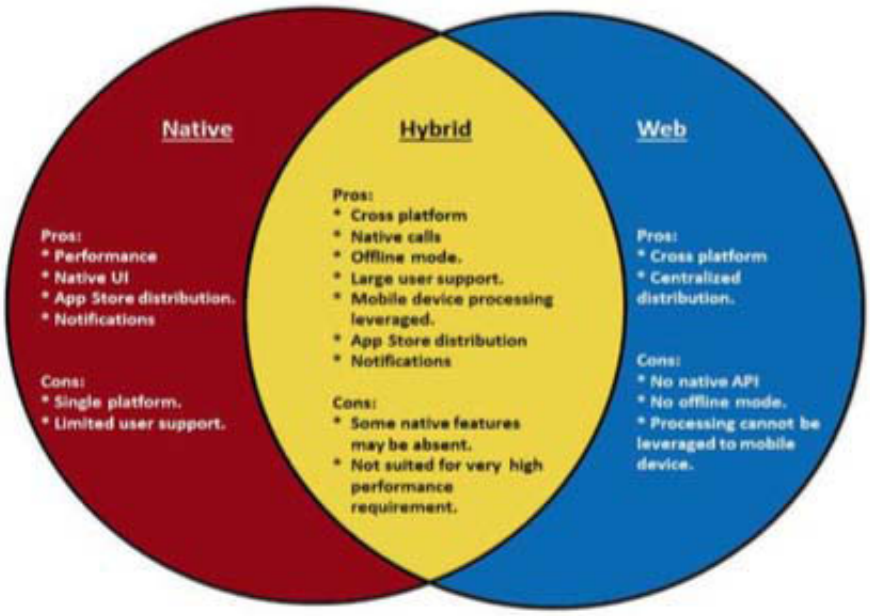


Figure 5.1: App development strategies according to [7]

to use. These projects are under fast development because these open-source projects are backed by big companies and big communities.

Xamarin, React Native and PhoneGap are free and reliable solutions and therefore very suitable for start-ups to use. The projects have the same goal, to provide a way to target multiple mobile platforms with a single code base, but the implementations are fairly different. This chapter will describe how the selected frameworks will be used in the comparison of this Thesis.

5.1 Development tools

The documentation that Google provides to write Android applications recommends to write Java code using Android Studio. We will use this as the standard environment for Android development. Writing Swift in XCode is the recommended environment to develop iOS applications and the standard instruments for Windows phone programming are C++ and Visual Studio. This Thesis will not investigate these environments in detail, but these environments are meant when referred to native development environments.

Xamarin provides a standalone application to develop apps: Xamarin Studio.

Xamarin Studio contains most of the features of Android Studio, XCode and Visual Studio. Android, iOS and Windows could be targeted without using any other tools for developing. React Native and PhoneGap are mainly based on web technologies as the only way to build apps is by coding in JavaScript. Therefore, any text editor could be used and no custom editor is provided. In both environments, it is recommended to start with a template project containing code for native apps. These native apps embed a JavaScript project that is going to be the custom code. The native projects function like wrappers and should barely be adapted. Table 5.1 shows an overview of the three cross-platform environments compared to the native environments.

	Native	Xamarin	React Native	PhoneGap
Targeted platforms:	Android, iOS, Windows	Android, iOS, Windows	Android, iOS	Android, iOS, Windows
Recommended editor	Android Studio, XCode, Visual Studio	Xamarin Studio	Text editor	Text editor
Recommended language	Java, Swift, C++	C#	JavaScript	JavaScript

Table 5.1: Recommended environments for different cross-platform development strategies

5.2 User interface design tools

User Interfaces are generally either declared in XML or are instantiated at runtime. During the compilation of an application, each XML layout file is compiled into a native UI resource. Listing 5.1 is an example of a basic Android user interface declaration. iOS and Windows development tools have comparable ways to declare interfaces in an XML related standard. There are, however, What You See Is What You Get (WYSIWYG) editors to create interfaces in a visual manner instead of declaring elements in XML. The WYSIWYG editors are platform specific as Android, iOS and Windows 10 have different XML definitions.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
</LinearLayout>
```

Listing 5.1: Example of an Android user interface defined in XML

In Xamarin, React Native and PhoneGap are the user interfaces defined in code only. Chapter 6 will describe how each framework came up with a shared user interface library for multiple platforms.

5.3 Prototype definition

Lecter's app is a simple app that mainly reads and edits data that is stored on a remote server. The graphic design has no strict requirements. Lecter needs to be developed fast and customizing the user interface is therefore no requirement at this stage.

The prototype will represent the main part of Lecter: viewing the available trainings and load a complete training after selecting it. The prototype consists of two main screens for these two tasks. Figure 5.2 shows an example of the interface that needs to be implemented to show the available trainings. This screen contains a grid of icons, representing the trainings, with a title. This screen also includes a search bar as a visual element without a search algorithm connected. After tapping one of the trainings, a new screen will appear containing the first slide or an input field to enter a password if the training is private (figure 5.3).

The prototypes must be able to support the following use cases:

- A user opens the app and sees the titles of the latest trainings that are fetched from the Lecter server. The search bar could be tapped to

insert a query to search for a training (without getting the results, as the search algorithm is not implemented on the server).

- A user taps one of the trainings. This will open the training and the first slide will be shown. Tapping the back button in a top bar results in a transition back to the home page containing an overview of the available trainings.

By creating this prototype, the following aspects of programming are implemented:

- Basic UI elements: Functional buttons, a search bar, labels and toolbars are used.
- Navigation: To switch between the home screen and the viewer screen, navigation is needed.
- Networking: The trainings are fetched from a server, so REST request need to be implemented.
- Business logic: Transforming a JSON object from the server into useful objects to view in the interface, is one of the aspects where basic programming is needed.

These different aspects are important when building a data retrieving app as Lecter. The cross-platform development frameworks can be compared based on these programming facets. Important consequences in the development process could be found by implementing the listed aspects and the ease of the implementation will indicate how suitable the frameworks are for building data retrieving apps.

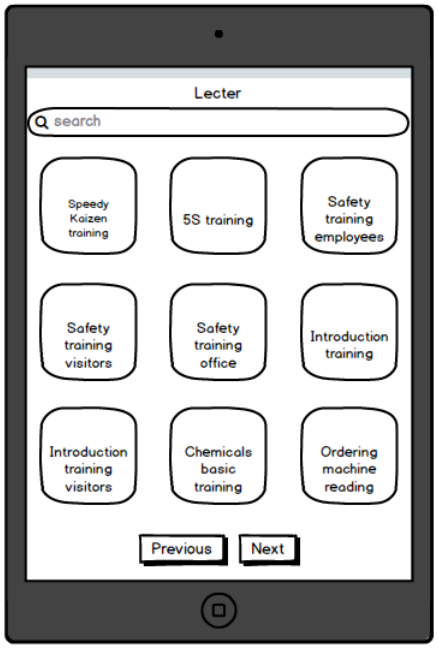


Figure 5.2: Mockup of prototype, showing the available trainings and a search bar

Figure 5.3: Mockup of prototype, showing a selected training and a back button

5.4 Research metrics

The frameworks are compared based on metrics and those metrics will be explained in this section. The metrics are related to business (section 5.4.1), the development (section 5.4.2) and the product (section 5.4.3).

5.4.1 Business metrics

To be able to select the best framework for Lecter, the impacts on the business of Lecter have to be taken into account. Lecter is a start-up now, but the possibilities to grow and the business impacts in case Lecter grows are also important factors when comparing the frameworks. The requirements for the development of Lecter in terms of business aspects are mainly that the development costs should be low and that at least iOS and Android could be targeted. Other requirements, that come in case Lecter grows, are that the product should be highly customizable and the development should not be platform dependent. The items to investigate concerning the business impacts are listed:

- **License model**: The costs of using a framework will be discussed. The consequences concerning the costs in case Lecter grows as a development team are also taken into account.
- **Supported OS**: The supported operating systems to target are compared. Android and iOS have to be targeted, but additional options are advantageous.
- **Supported workstation OS**: The availability of a software development kit is tested for the major operating systems: Windows, MacOS, Linux.
- **Visual customizability**: Lecter's app is mainly viewing or editing textual data. Data retrieving apps are usually able to be built out of standard user interface components. As a start-up, Lecter needs no unique visual style for branding reasons yet. The ability to create a customized interface might be interesting in a later stage, so this will be reviewed and compared between the different frameworks.

5.4.2 Development metrics

The way to develop an app depends heavily on the tools that are used. The selected frameworks are used according to the default environment with the

applications and tools that are recommended in the official documentation as specified before (table 5.1). It is important for Lecter that the used framework supports fast development, accepts third party libraries and will be continued to be supported in the next years. The aspects that affect the app development are listed in this section:

- **App compilation time**: The time to compile an app is measured in seconds. This time does not include the deployment time.
- **Deployment time**: The deployment time is measured in seconds. If no deployment is needed to view updates, for instance for debugging reasons, the time to refresh the app is measured.
- **Build time**: To measure the time to build a project, the project is cleaned first and the app is removed from the device. After that, the project will be built but not deployed. The logs that are written real-time, indicate when the project is started to be built and when it is built. The time between these two moments is measured and rounded to seconds multiple times to measure an average with a tighter confidence interval. The build time should give an indication of the relative build time compared to the other frameworks. The build time depends on many more aspects than the used framework and is therefore rounded to seconds. For instance: the specifications and background activities of both the development workstation and the targeted device affect the build time. The conditions of the workstations and devices are aimed to be identical during the tests.
- **Open-source**: The comparison of the frameworks includes a check if the frameworks are open-source. This enables the developer to contribute to the project and to add functionalities if needed. Open source projects are in some cases backed by big companies to gain social capital [16]. In that case experienced employees of those firms contribute to the development which increases the software quality. If a framework is open-source, the number of contributors wills be counted to get an impression of the community and the development speed.
- **Support for external libraries**: The possibility to add external libraries will be explored. The official documentation will be reviewed to know where to find libraries and how to add them to the project.

5.4.3 Product metrics

The impacts on the product performance are also measured. The used frameworks may not have a drastic influence in the app performance. The load time and execution time are compared:

- **App load time**: The time between opening the app and showing the complete functional user interface is measured and rounded in seconds.
- **App execution time**: The REST request that is executed in the prototype to retrieve trainings from the Lecter server takes some time. This time, including putting the retrieved data in the user interface, is measured in seconds and compared. The server response time is considered to be equal on average after repeated experiments.

Chapter 6

Results

The prototype is implemented according to the specifications in section 5.3, using Xamarin Forms, React Native and PhoneGap. The frameworks, default Software Development Kits (SDKs) and the built products are researched in this chapter in order to determine the consequences of using the selected frameworks.

6.1 Xamarin Forms

Xamarin is a mobile app development tool and is provided for free by Microsoft for small teams. Projects can target one of the following platforms: Android, iOS, macOS and Windows, or a combination of platforms with Xamarin Forms. Xamarin Forms, the tool to create cross-platform apps of Xamarin, is used to implement the prototype for the research in this Thesis. The software to build the applications, is Xamarin Studio or Visual Studio. These programs are free and available for Windows and Mac, but do not contain a visual interface builder for Xamarin Forms. Interfaces should be declared in XAML or coded in C#.

Start-ups have full access to the software without any limitations except that the development team has to consist of five developers maximum. There are many third party libraries and components available as a result of the open-source status. The community of independent developers is able to contribute to the development of Xamarin and that increases the robustness of the software. Contributing to Xamarin requires a membership. Developers can apply to join the open-source development community but the community is closed for people without the membership. External libraries are available and easy to find on components.xamarin.com.

The way to implement interfaces and how Xamarin converts it to native code will be explained in section 6.1.1. How the C# runs on the different platforms is explained in section 6.1.2. Section 6.1.3 illustrates the product performance impacts for Xamarin Forms apps.

6.1.1 Graphical User Interfaces

XAML or C# is used to make Graphical User Interfaces (GUIs) with Xamarin. A library of UI components is provided, like the iOS and Android operating systems do. Xamarin provides this library as a common standard for iOS, Android and Windows because these operating systems offer different UI frameworks. Table 6.1 shows possibilities to port the basic Xamarin components to native OS components.

	Xamarin	iOS	Android
Organize screen layout	Pages	ViewControllers	Activities
Structure view groups	Layouts	Constraints	ViewGroups
Basic UI components	Views	Views	Views
Reusable cells	Cells	CellViews	Views

Table 6.1: Xamarin User Interface component types related to iOS and Android User Interface components

An example of a Xamarin Forms user interface converted to a native iOS and Android interface is illustrated in figure 6.1. The parent object of a user interface in Xamarin code is always a subclass of Page but in iOS and Android it is a ViewController and a Activity respectively. Xamarin makes the conversion automatically so that iOS and Android applications run the native elements, but in a very similar way, as specified in the common standard of Xamarin Forms. The example given in the official documentation is a visual text element: a Xamarin Forms Entry becomes a UITextView on iOS and an EditText on Android.

User interface components, either defined in XAML or coded in C#, are interpreted during runtime. Listing 6.1 shows an example of a simple user interface containing a search bar only. At runtime, each page and its controls are mapped to platform-specific native user interface elements. This adds (claimed to be negligible) overhead. User interface components defined in XAML can be mapped during compilation time as well, XAMLC (XAML Compilation) has to be enabled to achieve this.

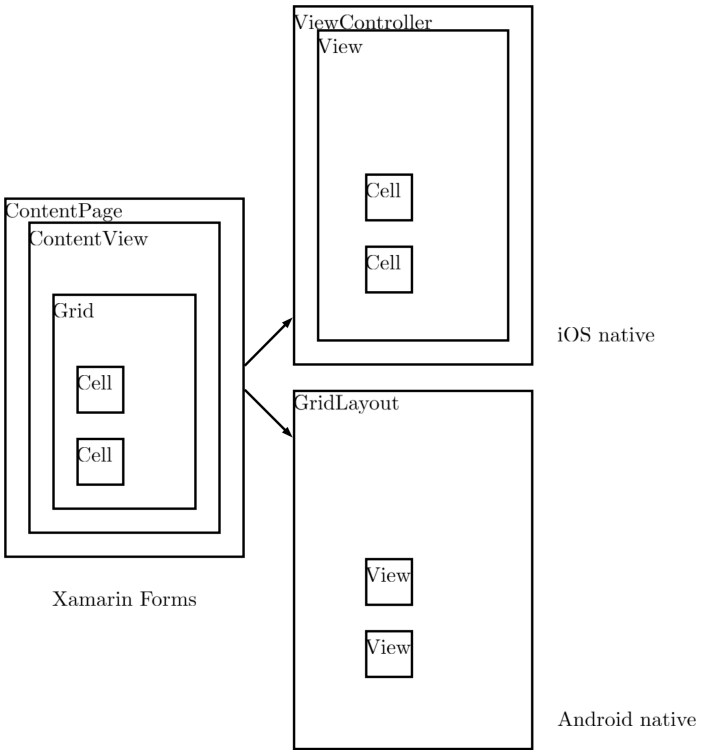


Figure 6.1: Example of Xamarin UI components ported to native iOS and Android components

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentView xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="LecterX.HomeView" xmlns:controls="clr-
    namespace:LecterX;assembly=LecterX">
  <ContentView.Content>
    <RelativeLayout>
      <SearchBar Placeholder="Search for Lecter" x:Name="
    LecterSearchBar"
        RelativeLayout.WidthConstraint =
          "{ConstraintExpression Type=RelativeToParent,
          Property=Width,
          Factor=1,
          Constant=0}"
        RelativeLayout.XConstraint =
          "{ConstraintExpression Type=RelativeToParent,
          Property=X,
          Factor=0,
          Constant=-0}"
        RelativeLayout.YConstraint =
          "{ConstraintExpression Type=RelativeToParent,
          Property=Y,
          Factor=0,
          Constant=-0}" />
    </RelativeLayout>
  </ContentView.Content>
</ContentView>
```

Listing 6.1: Example of a Xamarin user interface defined in XAML

6.1.2 Business logic

The code of Xamarin apps is written in C# but Android and iOS do not support C#. The strategy to make the C# code running on these platforms is different for both platforms.

On Android, Xamarin ships a fully functional C# runtime, called Mono, bundled with your app. This adds about 2.5MB to the app size, but enables the app to run C# on Android without being interpreted. This should barely impact the execution performance.

For iOS, Mono is used to map C# code to native iOS compatible code during app compilation. This increases the time to compile, but the performance is not compromised.

Xamarin provides mappings to all platform specific APIs, even APIs that are exclusive to one platform. The functionality to read fingerprints on iOS has no equivalent for Android, for example, but is yet available in Xamarin. This could result in little differences between the products for the targeted platforms if needed, as certain functionalities are only supported by one platform.

6.1.3 Performance

Compiling an app with Xamarin Forms takes relatively a long time. Building and deploying the app for the first time takes 30-45 seconds for Android and more than a minute on iOS. Rebuilding and deploying after small changes in the code takes 20-25 seconds in Android. On iOS it takes about the same time as building and deploying for the first time: more than a minute. In case no changes were made and no rebuild is needed, it takes only 2 seconds to start debugging in Android and 15 seconds in iOS.

Loading the app takes about 5 seconds which is more than the prototype built with React Native and PhoneGap. Further research would be interesting to investigate the reason of this long loading time.

Loading the trainings and adding them to the interface takes less than a second. Navigating to a new screen takes no extra delay as the navigation between screens is managed in the same way as natively developed apps. Table 6.2 shows an overview of the results of the measurements in seconds.

	Android	iOS
First time build and deploy	37 (±6)	83 (±10)
Build and deploy after code changes	24 (±2)	78 (±4)
Debugging without rebuild	2 (±0.5)	16 (±0.5)
App load time	5 (±0.5)	6 (±0.5)
Server fetch data and update UI	0.5 (±0)	0.5 (±0)

Table 6.2: Performance indicators for Xamarin development, means measured in seconds with sample standard deviation between brackets and n=10.

6.2 React Native

The goal of Facebook by developing React Native is clearly written in the documentation: "React Native enables you to build world-class application

experiences on native platforms using a consistent developer experience based on JavaScript and React." The developer experience is important and is always based on the React framework. As React lets developers design applications only according to the Flux architecture[11], using React Native has big implications in the general software architecture [12].

It is currently not supported to target Windows devices with React Native. The official documentation does not mention Windows as a possible platform to target, unlike Android and iOS. However, Windows announced recently a bypass to target Windows with React Native[1]. As React Native is a relatively new and fast growing project, Windows might be added in the near future as a optional platform to target. The project is open-source and free to use as a development tool. React Native is accessible on GitHub and has almost a thousand contributors.

JavaScript is used to write the applications. Some platform-specific boiler plate code is generated during the creation of a new project. This code consists of Objective-C code for the iOS product and Java code for the Android product, but needs not to be modified. Any text editor can be used to develop an app because the application specific code is written in JavaScript.

6.2.1 Graphical User Interfaces

The graphical interfaces are defined in XML, inside the JavaScript code. The visual elements are styled using CSS, which makes it easy to customize the layout of the app. The style names and values usually match how CSS works on the web, except names are written in lower camel case instead of hyphenated. There is no visual WYSIWYG editor to create user interfaces, interfaces will always be designed in code. Listing 6.2 shows a simple example of a declared interface.

[1]React Native on the Universal Windows Platform, 14 July 2016: https://blogs.windows.com/buildingapps/2016/04/13/react-native-on-the-universal-windows-platform/

```
class MainScene extends Component {
  render() {
    return (
      <View style={styles.myCell}>
        <Text style={styles.myText}>Hello World!</Text>
      </View>
    )
  }
}

var styles = StyleSheet.create({
  myCell: {
    color: 'blue',
  },
  myText: {
    textAlign: 'center',
  },
});
```

Listing 6.2: Example of a React Native user interface element in XML

6.2.2 Business logic

The JavaScript is interpreted during runtime, at least in development mode, which boosts the development speed. The app does not need to be recompiled after changing code as a simple reload will interpret the JavaScript code again. The work station that is used to develop the app runs a server and will connect to the device or simulator to serve the JavaScript code. Figure 6.2 shows the debug cycle, where step 1 only needs to be executed if the app is not deployed on the device yet. The native app that is installed in step 1 contains besides the provided standard React Native project, the app name, app icon and server address. XCode is needed when the deployment targets an iOS device. An app can be deployed from the command line when you are targeting Android or a simulator. Step 2 and step 3 are executed repeatedly during development. Step 3, reinterpreting the code, barely takes time compared to recompiling a native app.

6.2.3 Performance

As in debugging mode the code is stored on a server instead of the device, the server always has to be started. This takes around 30 seconds and can be done from the command line or automatically as part of the building process. The server runs to serve the Android app, the iOS app, or both.

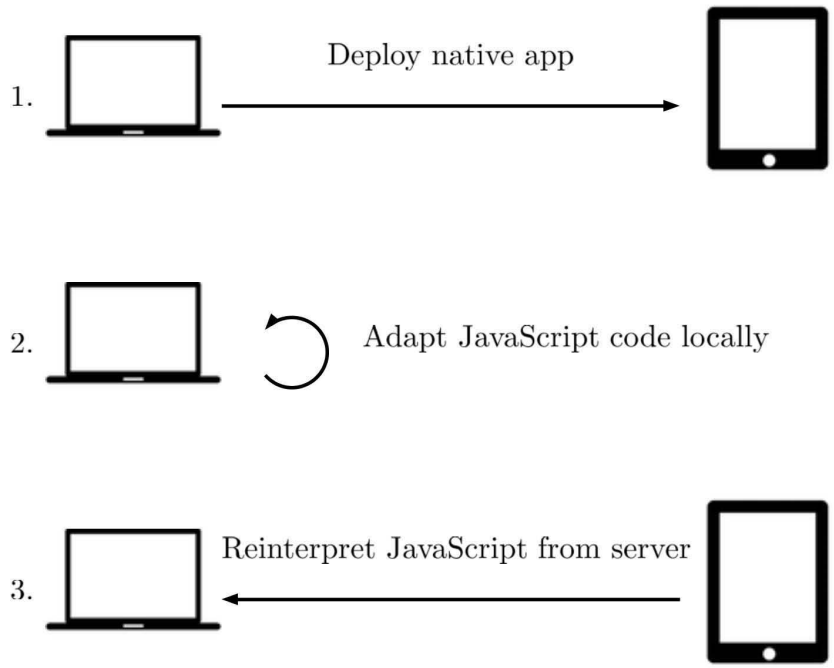


Figure 6.2: Debug cycle for app development using React Native

It takes about 50 seconds to build and deploy the app for the first time for Android and it can take up to 2 minutes when targeting iOS. After that, the code can be changed and reinterpreted quickly. Shaking the device triggers a reload and this takes only a couple of seconds.

Debugging can always be continued as the server never stops running. The app loading time is a couple seconds for Android and around 4 seconds for iOS. When the app is loaded it takes less than a second to retrieve the trainings from the Lecter server and put it in the user interface. Table 6.3 provides an overview of the measured times.

6.3 PhoneGap

Adobe PhoneGap is an app development environment that uses Apache Cordova. Apache Cordova is a free and open-source framework to target multiple platforms with one codebase. Hundreds of developers contribute to the many

	Android	iOS
Start server (OS independent)	31 (±2)	-
First time build and deploy	50 (±2)	99 (±21)
Build and deploy after code changes	2 (±0)	2 (±0)
Debugging without rebuild	0 (±0)	0 (±0)
App load time	2 (±0)	4 (±0.5)
Server fetch data and update UI	0.5 (±0)	0.5 (±0)

Table 6.3: Performance indicators for React Native development, means measured in seconds with sample standard deviation between brackets and n=10.

Cordova projects on GitHub.

iOS, Android, Windows and more platforms are supported and apps are built in HTML, CSS and JavaScript. Apps developed with PhoneGap will not run natively but in an embedded browser. The provided libraries give easy access to native functions as the camera and the device motion information [27]. This way, PhoneGap enables developers to build apps with the characteristics of a native app, but in a web development manner. The outputted apps will be available in the app stores and have access to device APIs, unlike web apps.

6.3.1 Graphical User Interfaces

User interfaces are written in HTML5 and will run in an embedded browser. The goal of PhoneGap is to make the app look native instead of a web page and removed therefore some browser functionalities. There will not be a visible search bar like a normal browser and text/image selection is disabled. An example of a simple user interface is provided in listing 6.3.

As the example in listing 6.3 shows, the app is built like a web page and a Cascading StyleSheet is used to style the elements. Navigating between screens is handled like hyperlinks. A new screen can be loaded, but there is no navigation like native apps. There are therefore no animated transitions between pages and there are no default navigation bars.

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="css/index.css" />
    <title>Hello World</title>
</head>

<body>
    <div class="app">
        <p>Hello world!</p>
    </div>
    <script type="text/javascript" src="cordova.js"></script>
    <script type="text/javascript" src="js/index.js"></script>
</body>

</html>
```

Listing 6.3: A simplified definition of a PhoneGap app user interface

6.3.2 Business logic

The app is mainly written in JavaScript and this will not be translated to an other language to be compiled. The JavaScript code will be interpreted at runtime. During the app development, the JavaScript code is edited and stored on the development workstation and a server will run, so that the compiled app on a test device can interpret the updated JavaScript code without a recompilation. This process is the same as the development process when using React Native. Figure 6.2 applies therefore to PhoneGap as well. When the app is built to be released, all files will be stored in the app so the server on the workstation is not needed anymore.

PhoneGap also provides wrappers for each targeted platform, like React Native. These projects contain mainly the app's metadata, like the app name and icons, and the mechanism to read the JavaScript from a server and publish it in the app in a hidden embedded browser. These wrapper projects are used when publishing the app. An app provided by PhoneGap via the app store is used during development. This app replaces the wrapper app and it only loads an app after submitting the server address. Using this app, a four-finger tap reloads the app to speed up debugging.

6.3.3 Performance

The first time build and deployment takes only 10-15 seconds. The server needs to run before building, but this takes only one second. While the server is running, debugging is enabled. Therefore there is no need to rebuild or reconnect to start debugging.

After the PhoneGap app is installed on the device, and the PhoneGap server is running on the development work station, debugging can be done very quickly. Reloading the app by tapping with 4 fingers takes about 12 seconds on Android and 5 seconds on iOS.

When the app is released and installed, it takes a few seconds to load the app. After the app is loaded, it takes less than a second to load the Lecter trainings from a server and to display it. Table 6.4 shows an overview of the measured data.

	Android	iOS
Start server (OS independent)	1 (±0)	-
First time build and deploy	15 (±1)	9 (±1)
Build and deploy after code changes	12 (±2)	4 (±2)
Debugging without rebuild	0	0
App load time	2 (±0)	4 (±0)
Server fetch data and update UI	0.5 (±0)	0.5 (±0)

Table 6.4: Performance indicators for PhoneGap development, means measured in seconds with sample standard deviation between brackets and n=10.

Chapter 7

Conclusion

To conclude this thesis the research question will be answered, including the subquestions:

- What are the differences between the suitable cross-platform mobile development environments for Lecter's front-end development?
  - What are the impacts on the development that come with each environment?
  - What are the impacts on the product that come with each environment?
  - What are the impacts on Lecter's business model that come with each environment?

7.1 Development impacts

The development cycles are very different. The frameworks use different programs and use different programming languages as specified in table 5.1.

Development with Xamarin is close to native Android and iOS development. Xamarin uses a Object Oriented programming language, like Android and iOS and Xamarin Studio has many similarities with Android Studio and XCode. The apps that are created with Xamarin are compiled to native apps and all native APIs are available. There is, however, no visual editor to create interfaces. Xamarin projects generally contain platform specific code because not all functionalities have a common Xamarin API. Therefore, only simple apps can be created with a single codebase.

React Native is slightly more like Web development. Projects are coded in

JavaScript and the visual elements are styled like how it is done in Web development: with CSS. React is a framework that forces developers to write apps in a Flux architecture. This is generally easier to learn for Web developers with experience in Angular, than for OOP developers who prefer to write traditional MVC structured programs. Debugging is fast because apps interpret JavaScript from a server, so recompilation is not needed when the code is changed.

PhoneGap is even more focused on Web development. From the developer's point of view, it is exactly Web development with pages written in HTML, elements styled with CSS and code written in JavaScript. Basic elements, for instance buttons, look therefore different. The lack of certain widely used native functions, like animated navigation bars and transitions, makes it harder to create a flawlessly functioning app. Debugging is fast because, like in React Native, the JavaScript is interpreted from a server and no recompilation is needed when the code has changed.

	Debug in browser	Default HTML5 development	Debug without recompilation	Standardized cross platform elements
Web app	Yes	Yes	Yes	Yes
PhoneGap	No	Yes	Yes	Yes
React Native	No	No	Yes	Yes
Xamarin Forms	No	No	No	Yes
Native	No	No	No	No

Table 7.1: Classification of Xamarin Forms, React Native and PhoneGap

Xamarin, React Native and PhoneGap could be placed on a spectrum between Web apps and native apps. Table 7.1 shows how this can be done based on some simple development characteristics. The app development is affected by the decision between the frameworks in terms of time, programming language and possibilities. A programmer should take these three impacts in consideration when choosing a cross-platform development tool.

7.2 Product impacts

Concerning the performance of the app, the apps are comparable for a simple data retrieving app as Lecter. Xamarin's app takes a bit longer to load, but the trainings are loaded and displayed quickly in all cases. PhoneGap provides less APIs to commonly used interface elements. Elements like buttons, check boxes and search bars look different from native elements and are not animated. It requires a lot of effort to make an app with PhoneGap that is as refined as native apps.

In all the three cases, the apps are downloadable via an app store and therefore offline accessible when it is installed. Apps created with PhoneGap have a different feel as the lack of native elements is noticeable. Table 7.2 shows some basic differences in the apps created with Xamarin, React Native and PhoneGap. Again, the frameworks could be placed on a spectrum between Web apps and native apps, where PhoneGap is close to Web development and Xamarin close to native development.

	Offline accessible	App contains native elements	Code is compiled	Single platform restriction
Web app	No	No	No	No
PhoneGap	Yes	No	No	No
React Native	Yes	Yes	No	No
Xamarin Forms	Yes	Yes	Yes	No
Native	Yes	Yes	Yes	Yes

Table 7.2: Classification of Xamarin Forms, React Native and PhoneGap

So for a basic data retrieving app, the visual impacts are more important than the performance in terms of speed or responsiveness. Frameworks closer to Web development are highly customizable, but using the native libraries from frameworks that are closer to native development results in more refined and standardized user interfaces. Therefore, the look and feel of the app is influenced by the choice between the selected framework.

7.3 Business impacts

Cross-platform development could speed up the development which leads to an earlier product release. Also, maintaining and updating the app could be faster, and more reliable as all targeted platforms share a identical codebase. These advantages are shared by Xamarin, React Native and PhoneGap.

Developers have to get familiar with each framework when using it for the first time. Web developers will easily adapt to PhoneGap, but developers without experience in React or any comparable framework like Angular, will have to learn how to use React Native. Developers with experience in native development will adapt easily to Xamarin Forms.

For small and simple apps, all frameworks are generally suitable. PhoneGap has benefits when visual customization is important, as CSS is an easy and powerful tool to do this. When apps are big and have to be very robust and refined, the lack of native elements is a big disadvantage for PhoneGap. Developers should therefore estimate if the product will get higher requirements in the future. In that case, PhoneGap would be less suitable compared to Xamarin Forms and React Native.

By choosing one of the three frameworks that are discussed in this Thesis, the business will be affected. If the product requirements become tighter in the future and the selected framework does not support these requirements, a stagnation in business growth might occur. The development speed is another business impact, as fast development implies earlier product releases.

7.4 Further research

Lecter is a small app with primarily basic function as fetching textual data from a server, display the information, and edit the data. This Thesis investigated some cross-platform development tools for this case, but other cases might lead to different results. How the frameworks handle big applications, or applications with much data processing, would be interesting to research. This further research should include more testing devices to get more reliable data.

The difference between native user interfaces and Web based interfaces are

visible but hard to measure. The exact differences in performance would be interesting to research further, to gain a clearer view on the exact differences.

The prototype that was implemented using Xamarin takes a longer time to load, compared to React Native and PhoneGap. The reason for this is not found in this Thesis. This result was unexpected and could be subject for further research.

7.5 Recommendations

Lecter is going to be an app with few functionalities that has to be available at Android and iOS. In combination with the low budget and the desire to publish soon, a cross-platform development tool is very useful.

In case Lecter evolves and gets higher requirements, PhoneGap will not be the right solution. The app will not feel as solid as other professional apps to the users. PhoneGap is useful to build a prototype, but it will be too hard to meet the expected future requirements. The lack of support for native libraries results in less refined user interface elements. Developing Lecter with PhoneGap might therefore result in a switch to an other cross-platform development tool in the future. In that case, the app has to be redeveloped from scratch because other frameworks use different programming environments and different architecture styles. PhoneGap is therefore not recommended to use for the product development of Lecter.

I found React Native and Xamarin Forms to be suitable even if Lecter evolves. Both frameworks are very complete and yet under development to improve. Both frameworks are widely used and supported by big companies, so the online documentation will grow and the development/support is expected to continue for a long time. These frameworks might even be the future in mobile development, and outgrow native development.

The biggest difference for Lecter between React Native and Xamarin Forms, is the development style. React Native projects are written in JavaScript and Xamarin Forms projects in C#. For developing apps like Lecter, I recommend one of these frameworks based on the skill set of the development team. In the case of Lecter, the development team is already familiar with native development but has not much experience with Web development. Xamarin Forms has slightly more APIs to native functions, but React Native is slightly easier to customize visually. The app performance is in Lecter's case about

equal with React Native and Xamarin. The development style is therefore decisive thus Xamarin is the recommended framework to develop Lecter.

All frameworks can target both iOS and Android but are different in the approach to do that. The decision between the frameworks is made based on the expected future requirements and the skill set of the development team. Xamarin Forms is therefore the most suitable cross-platform development environment for Lecter.

Bibliography

[1] Agarwal, S., Goswami, S., and Nath, A. Green computing and green technology in e-learning, corporate, business and it sectors. *International Journal of Computer Applications 76*, 7 (2013).

[2] Batalla-Busquets, J.-M., and Pacheco-Bernal, C. On-the-job e-learning: Workers' attitudes and perceptions. *The International Review of Research in Open and Distributed Learning 14*, 1 (2013), 40–64.

[3] Charland, A., and Leroux, B. Mobile application development: web vs. native. *Communications of the ACM 54*, 5 (2011), 49–53.

[4] Clark, R., and Mayer, R. *e-Learning and the Science of Instruction: Proven Guidelines for Consumers and Designers of Multimedia Learning*. An essential knowledge resource. Wiley, 2011.

[5] Curtin, M. Write once, run anywhere: Why it matters. *Technical Article. http://java. sun. com/features/1998/01/wo* (1998).

[6] Dale, E. The cone of experience. *Audio-visual methods in teaching 1* (1946), 37–51.

[7] de Andrade, P. R., Albuquerque, A. B., Frota, O. F., Silveira, R. V., and da Silva, F. A. Cross platform app: a comparative study. *arXiv preprint arXiv:1503.03511* (2015).

[8] Downes, S. E-learning 2.0. *Elearn magazine 2005*, 10 (2005), 1.

[9] Essalmi, F., Ayed, L. J. B., Jemni, M., Graf, S., et al. Generalized metrics for the analysis of e-learning personalization strategies. *Computers in Human Behavior 48* (2015), 310–322.

[10] Feng, X., Shen, J., and Fan, Y. Rest: An alternative to rpc for web services architecture. In *Future Information Networks, 2009. ICFIN 2009. First International Conference on* (2009), IEEE, pp. 7–10.

[11] Fisher, B. Flux: A unidirectional data flow architecture for react apps. In *Applicative 2015* (New York, NY, USA, 2015), Applicative 2015, ACM, pp. –.

[12] Gackenheimer, C. Introducing flux: An application architecture for react. In *Introduction to React*. Springer, 2015, pp. 87–106.

[13] Gibbs, S. Flash is dying a death by 1,000 cuts, and that's a good thing. *The Guardian* (2015).

[14] Govindasamy, T. Successful implementation of e-learning: Pedagogical considerations. *The internet and higher education 4*, 3 (2001), 287–299.

[15] Haney, B. D. Assessing organizational readiness for e-learning: 70 questions to ask. *Performance improvement 41*, 4 (2002), 10–15.

[16] Homscheid, D., Schaarschmidt, M., and Staab, S. Firm-sponsored developers in open source software projects: a social capital perspective. *Twenty-Fourth European Conference on Information Systems (ECIS)* (2016).

[17] Jones, E. R. Implications of scorm and emerging e-learning standards on engineering education. In *Proceedings of the 2002 ASEE Gulf-Southwest Annual Conference* (2002), pp. 20–22.

[18] Joorabchi, M. E., Mesbah, A., and Kruchten, P. Real challenges in mobile app development. In *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (2013), IEEE, pp. 15–24.

[19] Leem, J., and Lim, B. The current status of e-learning and strategies to enhance educational competitiveness in korean higher education. *The International Review of Research in Open and Distributed Learning 8*, 1 (2007).

[20] Mayer, R. E., and Moreno, R. Nine ways to reduce cognitive load in multimedia learning. *Educational psychologist 38*, 1 (2003), 43–52.

[21] Mikkonen, T., and Taivalsaari, A. Apps vs. open web: The battle of the decade. In *Proceedings of the 2nd Workshop on Software Engineering for Mobile Application Development* (2011), pp. 22–26.

[22] Murugesan, S. Understanding web 2.0. *IT professional 9*, 4 (2007), 34–41.

[23] Naylor, D., Finamore, A., Leontiadis, I., Grunenberger, Y., Mellia, M., Munafò, M., Papagiannaki, K., and Steenkiste, P. The cost of the s in https. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies* (2014), ACM, pp. 133–140.

[24] Nneka Eke, H. The perspective of e-learning and libraries in africa: challenges and opportunities. *Library Review 59*, 4 (2010), 274–290.

[25] Osterwalder, A., and Pigneur, Y. *Business Model Generation: A Handbook for Visionaries, Game Changers*. Alexander Osterwalder & Yves Pigneur, 2009.

[26] Ousterhout, J. K. Scripting: Higher level programming for the 21st century. *Computer 31*, 3 (1998), 23–30.

[27] Pierre, N. J., and Octavien, M. Review of phonegap apis accessing the native mobile platform apis. *Lecture Notes on Software Engineering 4*, 1 (2016), 12.

[28] Seels, B. The relationship of media and isd theory: The unrealized promise of dale's cone of experience., 1997.

[29] Serrano, N., Hernantes, J., and Gallardo, G. Mobile web apps. *IEEE software 30*, 5 (2013), 22–27.

[30] Sharma, K. Financial implications of implementing an e-learning project. *Journal of European Industrial Training 35*, 7 (2011), 658–686.

[31] Sun, P.-C., Tsai, R. J., Finger, G., Chen, Y.-Y., and Yeh, D. What drives a successful e-learning? an empirical investigation of the critical factors influencing learner satisfaction. *Computers & education 50*, 4 (2008), 1183–1202.

[32] Suppes, P., Jerman, M., and Groen, G. Arithmetic drills and review on a computer-based teletype. *The Arithmetic Teacher* (1966), 303–309.

[33] Tai, L. Corporate e-learning: How e-learning is created in three large corporations. *Dissertations available from ProQuest* (2005).

[34] Violante, M. G., and Vezzetti, E. Virtual interactive e-learning application: An evaluation of the student satisfaction. *Computer Applications in Engineering Education 23*, 1 (2015), 72–91.

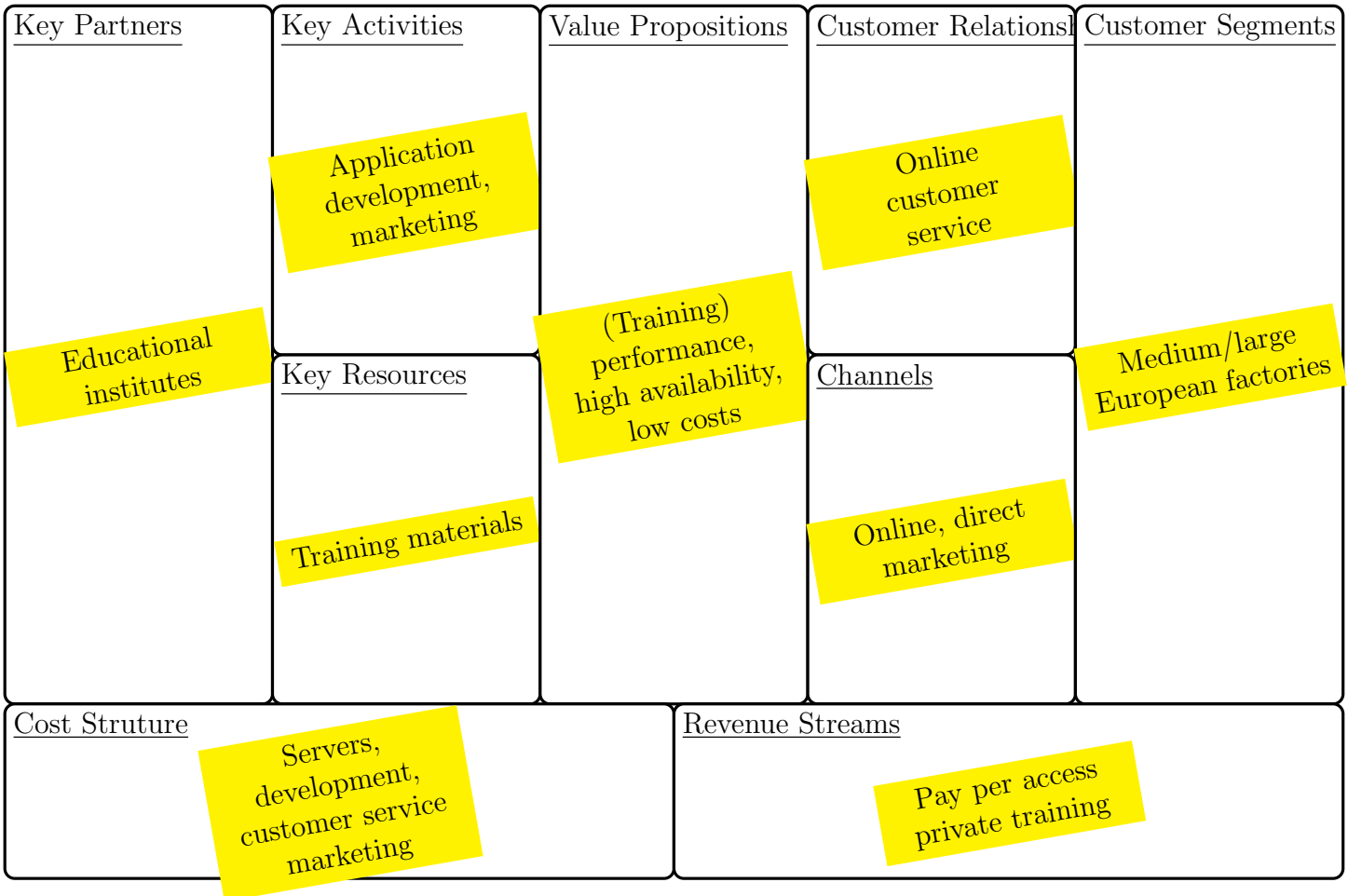
[35] Welsh, E. T., Wanberg, C. R., Brown, K. G., and Simmering, M. J. E-learning: emerging uses, empirical results and future directions. *International Journal of Training and Development 7*, 4 (2003), 245–258.

[36] Wu, Tsai, C. W. An integrative model to predict the continuance use of electronic learning systems: Hints for teaching. *International Journal on ELearning 5*, 2 (2006), 287.

Appendix A

Business Model Canvas of Lecter

This Business Model Canvas visualizes how Lecter aims to be a profitable company.



Appendix B

Used devices

The used workstation to develop the apps (and run a server in case of React Native and PhoneGap development) is a MacBook Pro. When deploying, debugging and measuring performance metrics an iOS and an Android tablet are used. Herewith the relevant product details:

Workstation:
Name: MacBook Pro (13-inch, Mid 2012)
Chip: 2,5 GHz Intel Core i5
Memory: 4 GB 1600 MHz DDR3
Disk: Macintosh HD

iOS tablet:
Name: iPad (3rd generation)
Chip: Dual-core Apple A5X
Storage: 16 GB

Android tablet:
Name: Iconia One 10
Chip: Cortex A53
Memory: 1 GB
Storage: 16 GB